**OpenFaaS调研分析**

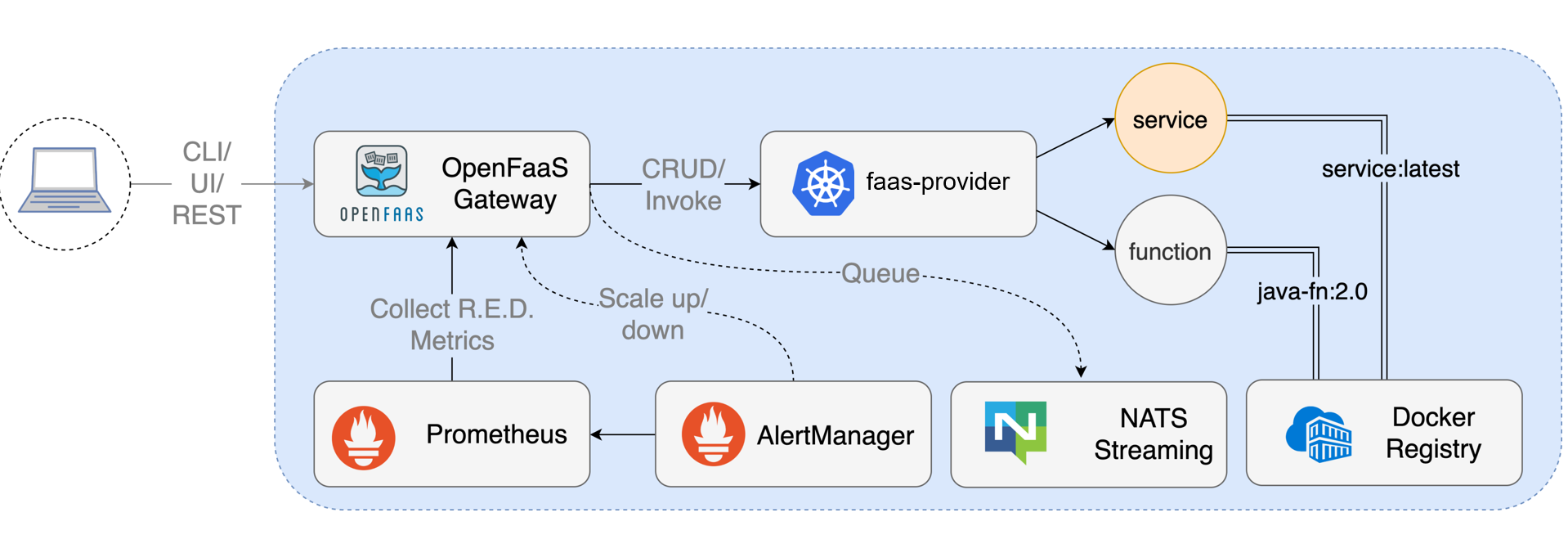
1. **OpenFaaS简介**

Serverless的全称是Serverless computing无服务器运算，又被称为函数即服务（Function-as-a-Service，缩写为 FaaS），是云计算的一种计算范式。以平台即服务（PaaS）为基础，无服务器运算提供一个微型的架构，终端客户不需要部署、配置或管理服务器服务，代码运行所需要的服务器服务皆由云端平台来提供。

OpenFaaS是一个开源的事件驱动的serverless框架, 为用户提供FaaS计算平台. OpenFaaS当前官方的支持都是基于docker容器的运行, 可以直接利用Kubernetes, Swarm等编排系统进行管理, 也可以利用faasd在资源有限的单机环境下执行.

OpenFaaS框架的整体设计与micropos非常类似, 每个功能都单独作为一个微服务, 对外提供REST API, 并且使用gateway进行转发. Openfaas是一个开放的框架, 对于后端如何运行云函数可以有多个不同的实现. 当前为了强隔离与缩放的因素, 最主流的都是利用容器, 每个函数都是在容器里面运行, 并且也是作为微服务的组件. 这些容器的管理编排工作主流的是使用Kubernetes来进行管理, 相应的管理部分faas-netes也构成了OpenFaaS整个框架的核心. Faas-provider只规范了函数管理的接口, 这就极大地增加了整个框架的灵活性, 不仅能利用容器, 还可以使用其他隔离方案来部署运行云函数.

1. **OpenFaaS架构**



图一: OpenFaaS workflow

OpenFaaS 架构如图一所示, 主要分为三个大部分:

Gateway: 提供一系列用于函数管理, 运行指标记录, 容器缩放的REST API, 并且将其转发给相应的微服务组件.与spring cloud gateway功能差不多, 能够根据配置提供网关服务.

CLI/UI: 内置的ui和faas-cli负责为用户提供管理接口,并向gateway发送对应的请求

Faas-provider: 提供一系列管理部署调用函数的REST API, faas-provider准确说是一个接口规范, 具体使用哪些编排工具是需要具体的实现, 如最流行的官方实现是faas-netes, 是利用kubernetes来编排容器进行管理函数.

另外openfaas还集成了其他工具, NATS负责异步函数的执行与消息通信,Prometheus负责运行指标记录, 性能监控以及自动缩放等

OpenFaaS的工作流: 用户通过REST API, 如利用faas-cli或者内置的UI来访问gateway, 然后gateway转发到相应的微服务组件中. 每个微服务组件有默认的端口路由, 也可以通过自定义来设置. Prometheus 通过gateway收集运行的指标, 监控性能, 并且根据相关条件发出自动缩放的请求.利用gateway转发相应的函数部署, 调用请求. 函数调用可以使用NATS流的队列进行异步返回. Faas-provider当前最主要的是使用faas-netes, 利用kubernetes进行容器编排, 控制函数的部署以及调用.

1. **FaaS-provider**

FaaS-provider是用go编写的SDK, 利用HTTP REST API与Gateway进行交互通信, 并与相关的接口兼容. 每个faas-provider接口都需要有以下的行为规范:

1. 函数的CRUD
2. 调用运行函数
3. 接收缩放指令, 通过缩放容器等行为缩放函数实例.
4. 密钥的CRUD以及日志流管理

当前Faas-provider的主要官方实现:

1. Faas-netes: 与Kubernetes容器编排服务集成, 使OpenFaaS能够通过docker, Kubernetes提供完整的无服务器功能支持. 与Kubernetes生态系统进行集成, 也能通过kubectl进行管理, 并且提供内置的UI, 可用的商业支持, 异步调用等功能.
2. Faasd: 相比于faas-netes对于集群的支持, faasd面向的是单个节点资源有限的机器. 能够在单个主机(如树莓派)下提供相同的功能(容器支持). Faasd通过修改contained和利用CNI, 使容器的部署执行比docker更节省资源, 更能适合serverless场景.
3. Faas-swarm: 利用docker swarm编排工具实现的容器管理, 函数管理.
4. **Watchdog**

Openfaas中最重要的是watchdog. 每个云函数都作为一个HTTP微服务运行, watchdog就是作为云函数微服务的反向代理, 将http的request内容作为函数的stdin, 将函数输出的stdout作为http的response返回. 或者将http请求转发到函数监听的http端口. 同时watchdog还利用Prometheus来监控当前性能, 并且返回相应的metrics.

实际的部署过程中, watchdog与相应的云函数在一起作为一个容器镜像, 运行的时候watchdog监听http请求, 并且启动相应的云函数进程. 云函数可以使用任意语言, 只要是可以运行的二进制文件即可.

Watchdog在执行函数的时候, 可以用多种模式来运行. 主要模式是http模式和Serializing模式. 对于Http模式, 将用fork启动云函数进程, watchdog将http请求转发给云函数相应的端口. Serializing模式也是利用fork启动云函数进程, 利用管道与云函数的进程通信, 重定向云函数的stdio. 模式的选用根据函数的功能来选择.

1. **Autoscaling**

自动缩放可以自动根据负载调整函数的实例副本的数量. 自动缩放也是实现了多种模式:

RPS: 根据每秒完成请求的个数, 适用于非常短时间的函数.

Capacity: 容量, 根据当前正在执行的个数, 适用于长时间的函数.

CPU: 根据当前cpu的负载来决定是否缩放.